GMMAT: Generalized linear Mixed Model Association Tests
Version 0.6

Han Chen
Department of Biostatistics
Harvard T.H. Chan School of Public Health
Email: hanchen@hsph.harvard.edu

Matthew P. Conomos
Department of Biostatistics
University of Washington

October 12, 2015

Contents

1 Introduction **3**

2 The model **3**

3 Getting started **3**
3.1 Downloading *GMMAT* 3
3.2 Installing *GMMAT* 4

4 Input **4**
4.1 Phenotype and covariates 4
4.2 Matrices of covariance structure 5
4.3 Genotypes 5

5 Running *GMMAT* **6**
5.1 Fitting GLMM 6
5.2 Score tests 8
5.3 Wald tests 9

6 Output **10**

7 Advanced options **11**
7.1 Alternative model fitting algorithms 11
7.2 Changing model fitting parameters 11
7.3 Missing genotypes 12
7.4 Reordered genotypes 12
7.5 Other options 12

8 Version **13**
8.1 Version 0.6 (October 12, 2015) 13

9 Contact **13**

10 Acknowledgments **13**

1 Introduction

GMMAT is an R package for performing association tests using generalized linear mixed models (GLMMs)[1] in genome-wide association studies (GWAS). GLMMs provide a broad range of models for correlated data analysis. In the GWAS context, examples of correlated data include those from family studies, samples with cryptic relatedness and/or shared environmental effects, as well as samples generated from complex sampling designs. GLMMs may also be used to account for population structure in GWAS.

GMMAT first fits a GLMM with covariate adjustment and random effects to account for population structure and family or cryptic relatedness, and then performs score tests for each genetic variant in the GWAS. For candidate gene studies, *GMMAT* can also perform Wald tests to get the effect size estimate for each genetic variant.

2 The model

In the context of single variant test, *GMMAT* works with the following GLMM

$$\eta_i = g(\mu_i) = \mathbf{X}_i\boldsymbol{\alpha} + G_i\beta + b_i.$$

We assume that given the random effects $\mathbf{b}$, the outcome y_i are conditionally independent with mean $E(y_i|\mathbf{b}) = \mu_i$ and variance $Var(y_i|\mathbf{b}) = \phi a_i^{-1} v(\mu_i)$, where ϕ is the dispersion parameter (for binary and Poisson data $\phi = 1$), a_i are known weights, and $v(\cdot)$ is the variance function. The linear predictor η_i is a monotonous function of the conditional mean μ_i via the link function $\eta_i = g(\mu_i)$. $\mathbf{X}_i$ is a $1 \times p$ row vector of covariates for subject i, $\boldsymbol{\alpha}$ is a $p \times 1$ column vector of fixed covariate effects including the intercept, G_i is the genotype of the genetic variant of interest for subject i, and β is the fixed genotype effect. We assume that $\mathbf{b} \sim N(0, \sum_{k=1}^{K} \tau_k \mathbf{V}_k)$ is an $n \times 1$ column vector of random effects, τ_k are the variance component parameters, $\mathbf{V}_k$ are known $n \times n$ matrices. In practice, $\mathbf{V}_k$ can be the theoretical kinship matrix if analyzing family samples with known pedigree structure in a homogeneous population, or the empirical genetic relationship matrix (GRM) to account for population structure and cryptic relatedness, or any $n \times n$ matrices to account for shared environmental effects or complex sampling designs.

GMMAT can be used to analyze both continuous and binary traits. For continuous traits, if a normal distribution and an identity link function are assumed, *GMMAT* performs association tests based on linear mixed models (LMMs). For binary traits, however, we showed that performing association tests based on LMMs can lead to invalid P values in the presence of moderate or strong population stratification, even after adjusting for top ancestry principal components (PCs) as fixed effects.[2] In such scenarios, we would recommend assuming a Bernoulli distribution and a logit link function for binary traits, adjusting for top ancestry PCs as fixed-effect covariates. This GLMM is also known as the logistic mixed model.

3 Getting started

3.1 Downloading *GMMAT*

GMMAT can be downloaded at `http://www.hsph.harvard.edu/xlin/software.html#gmmat`. It can be installed as a regular R package on a UNIX operating system. Currently we do not support Mac or Windows versions.

3.2 Installing *GMMAT*

Before installing *GMMAT*, please check your system to make sure C++ libraries boost (`http://www.boost.org/`) and Armadillo (`http://arma.sourceforge.net/`) have been installed appropriately. *GMMAT* also requires R packages Rcpp and RcppArmadillo, which can be downloaded from CRAN (`https://cran.r-project.org/web/packages/Rcpp/index.html` and `https://cran.r-project.org/web/packages/RcppArmadillo/index.html`).

To install *GMMAT* into "your_directory", in an R session you can use

```
> install.packages("GMMAT_0.6.tar.gz", lib = "your_directory")
```

If you do not have Rcpp or RcppArmadillo installed yet, you can use the following command to install RcppArmadillo directly from CRAN without downloading the package manually (before installing *GMMAT*), and Rcpp will be installed automatically.

```
> install.packages("RcppArmadillo", lib = "your_directory",
          repos = "http://cran.us.r-project.org")
```

4 Input

GMMAT requires the phenotype and covariates in an R data frame, known $n \times n$ matrices $\mathbf{V}_k$ as an R matrix (in the case of single matrix) or an R list (in the case of multiple matrices), and genotypes saved in a plain text file (or in a compressed plain text file .gz or .bz2), or in a PLINK binary file. We describe how to prepare these data below.

4.1 Phenotype and covariates

Phenotype and covariates should be either saved as a data frame in R, or recorded in a text file that can be read into R as a data frame. The rows of the data frame represent different individuals, and the columns represent different variables. For example, here we show the header and first 6 rows of the example text file pheno.txt:

```
disease trait   age     sex
1       5.45    61      0
1       5.61    50      1
0       3.1     54      0
1       6.22    48      1
1       5.42    49      0
0       6.22    50      1
...
```

In this example, there are one binary phenotype (disease), one quantitative phenotype (trait) and two covariates (age and sex). There can be additional columns for unused variables, and the order of columns does not matter. To read it into R as a data frame, you can use

```
> pheno <- read.table("pheno.txt", header = TRUE)
```

Missing values in the data frame should be recognizable by R as NA. For example, if you use . (period) to denote missing values in the text file, you can use

```
> pheno <- read.table("pheno.txt", header = TRUE, na.strings = ".")
```

4.2 Matrices of covariance structure

GMMAT requires at least one $n \times n$ matrix $\mathbf{V}_k$ to model the covariance structure of the random effects. In the simplest case, this is usually a GRM estimated from the genotype data. Currently *GMMAT* does not provide a function to calculate the GRM, but there are many software packages that can do this job. For example, GEMMA[3] can be used to estimate either the centered GRM or the standardized GRM. GRM saved in an external file must be read into R as a matrix. For example,

```
> GRM <- as.matrix(read.table("GRM.txt.bz2"))
```

Multiple $n \times n$ matrices can be used to allow multiple components of random effects. In such cases, the matrices should be constructed as a list of $n \times n$ matrices. For example, if you have 3 R matrices Mat1, Mat2 and Mat3, you can construct the R list

```
> Mats <- list(Mat1, Mat2, Mat3)
```

All matrices must be positive semi-definite. Before running the analysis, it is very important to make sure that the dimensions of these matrices must match the row number of the phenotype data frame, and the order of individuals in the rows and columns of these matrices must also match the order of individuals in the phenotype data frame. If not, it is usually easier to sort the phenotype data frame to match the order of individuals in the matrices.

4.3 Genotypes

GMMAT can take genotype files either in plain text format (or the compressed version .gz or .bz2), or in PLINK binary format. Non-integer imputed genotypes (dosages) should be saved in plain text files (or the compressed version .gz or .bz2). The plain text file can be space-, tab-, comma-, or even special character-delimited, and there can be additional rows (e.g., comments) and/or columns before the genotype data matrix. Here is an example of part of a tab-delimited plain text genotype file geno.txt:

```
# This is an example genotype file for demonstrating GMMAT
# Each row represents one SNP for all individuals in the study
# First column is SNP name, second and third columns are alleles (Allele1
and Allele2): it is recommended to use Allele1 for the reference allele
and Allele2 for the effect allele, but reversed coding is also allowed
and does not affect association test results (users should be cautious
with allele coding when interpreting results)
# Starting from fourth column, each column represents one individual
# In this example, there are 400 individuals and 100 SNPs
SNP1    A       T       0       0       NA      NA      ...
SNP2    A       C       1       0       1       0       ...
SNP3    A       C       0       0       0       1       ...
SNP4    A       G       1       0       1       1       ...
SNP5    A       G       1       0       2       1       ...
...
```

5 Running *GMMAT*

If *GMMAT* has been successfully installed into "your_directory", you can load it in an R session using

```
> library(GMMAT, lib.loc = "your_directory")
```

We provide 5 functions in *GMMAT*: **glmmkin** for fitting the GLMM with known $\mathbf{V}_k$, **glmm.score.text** and **glmm.score.bed** for running score tests, **glmm.wald.text** and **glmm.wald.bed** for running Wald tests. Details about how to use these functions, their arguments and returned values can be found in the R help document of *GMMAT*. For example, to learn more about **glmmkin**, in an R session you can type

```
> ?glmmkin
```

5.1 Fitting GLMM

Here we provide a simple example of fitting GLMM using **glmmkin**. We have the binary phenotype disease, and two covariates age and sex, saved in a plain text file pheno.txt. We also have computed the GRM externally and saved it in a compressed file GRM.txt.bz2, and have checked to make sure the order of individuals in the GRM matches the order of individuals in the phenotype data (VERY IMPORTANT). In this example we fit a GLMM assuming Bernoulli distribution of the phenotype and logit link function (also known as a logistic mixed model). We adjust for age and sex, and use one $n \times n$ matrix as $\mathbf{V}_k$ (the GRM) to model the covariance structure of the random effects.

```
> pheno <- read.table("pheno.txt", header = TRUE)
> GRM <- as.matrix(read.table("GRM.txt.bz2"))
> model0 <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"))
> model0$theta
[1] 1.0000000 0.3377318
> model0$alpha
[1]  0.472081666 -0.006818636 -0.086444753
> model0$cov
             [,1]          [,2]          [,3]
[1,]  1.21381858 -0.0228770449 -0.0416219165
[2,] -0.02287704  0.0004506977  0.0003935442
[3,] -0.04162192  0.0003935442  0.0436495422
```

Note that pheno and GRM must be read into R as a data frame and a matrix, respectively. When using the function **glmmkin**, the data frame of phenotype and covariates (in our example, pheno) should be passed to the argument "data", and the matrix or the list of matrices for random effects (in our example, the matrix GRM) should be passed to the argument "kins". The first argument of the function **glmmkin** is "fixed", which requires a formula for fixed effects. The syntax of the formula is the same as the formula used in a linear model **lm** and a generalized linear model **glm**. The example model above is equivalent to

```
> model0 <- glmmkin(fixed = disease ~ age + sex, data = pheno,
+         kins = GRM, family = binomial(link = "logit"))
```

The argument "family" takes the same syntax as used in a generalized linear model **glm**. For example, if you would like to fit a LMM for a quantitative trait, you can use

```
> model0q <- glmmkin(fixed = trait ~ age + sex, data = pheno,
+         kins = GRM, family = gaussian(link = "identity"))
```

Please avoid using LMMs for binary traits.

Here is a list of supported **family** objects (for details and alternative link/variance functions, please see the R help document of **family**):

Family	Link	Trait	Variance
binomial	logit	binary	$\mu(1-\mu)$
gaussian	identity	continuous	ϕ
Gamma	inverse	continuous	$\phi\mu^2$
inverse.gaussian	1/mu^2	continuous	$\phi\mu^3$
poisson	log	count	μ
quasi	identity	continuous	ϕ
quasibinomial	logit	binary	$\phi\mu(1-\mu)$
quasipoisson	log	count	$\phi\mu$

The function **glmmkin** returns a list. The first element in the vector theta is the estimate of the dispersion parameter ϕ (for binary and Poisson data we have a fixed $\phi = 1$), and the remaining elements are variance component estimates for each matrix modeling the covariance structure of the random effects (in the same order as in the list of matrices passed to "kins"). In our binary disease example above, we have only one such matrix (the GRM), and the results show that the estimate of the variance component parameter τ_k is 0.3377318. The vector alpha gives the fixed effect estimates, and the order matches the order of covariates in the formula passed to "fixed". In our binary disease example above, we have an intercept 0.472081666, and an age effect estimate -0.006818636, a sex effect estimate -0.086444753. The matrix cov is the covariance matrix of the fixed effect estimates.

In the following example, we have the same binary phenotype disease, covariates age and sex, and the same GRM as in the previous example. In addition to the GRM, we have another $n \times n$ matrix to model the covariance structure of the random effects. The GLMM is

$$\log(\frac{P(disease_i = 1|age_i, sex_i, b_i)}{1 - P(disease_i = 1|age_i, sex_i, b_i)}) = \alpha_0 + \alpha_1 \times age_i + \alpha_2 \times sex_i + b_i,$$

where $\mathbf{b} \sim N(0, \tau_1\mathbf{V}_1 + \tau_2\mathbf{V}_2)$, $\mathbf{V}_1$ is the GRM, $\mathbf{V}_2$ is a block diagonal matrix with block size 10 and all entries equal to 1 within a block. Here $\mathbf{V}_2$ is used to model clusters: in this example we have 40 clusters with 10 individuals in each cluster. For individual i in cluster j ($j = 1, 2, \ldots, 40$), the model above is equivalent to

$$\log(\frac{P(disease_i = 1|age_i, sex_i, b_{1i}, b_{2j})}{1 - P(disease_i = 1|age_i, sex_i, b_{1i}, b_{2j})}) = \alpha_0 + \alpha_1 \times age_i + \alpha_2 \times sex_i + b_{1i} + b_{2j},$$

where $\mathbf{b}_1 \sim N(0, \tau_1\mathbf{V}_1)$, $b_{2j} \sim N(0, \tau_2)$. All 10 individuals in a cluster share a common b_{2j}, and b_{2j} are independent and identically distributed across clusters.

```
> M10 <- matrix(0, 400, 400)
> for(i in 1:40) M10[(i-1)*10+(1:10), (i-1)*10+(1:10)] <- 1
> Mats <- list(GRM, M10)
> model10 <- glmmkin(fixed = disease ~ age + sex, data = pheno,
+        kins = Mats, family = binomial(link = "logit"))
> model10$theta
[1] 1.0000000 0.2199089 0.1219592
```

In this example, the dispersion parameter ϕ is fixed to 1, the variance component estimates $\hat{\tau}_1 = 0.2199089$, $\hat{\tau}_2 = 0.1219592$, corresponding to the $n \times n$ matrices $\mathbf{V}_1$ (the GRM) and $\mathbf{V}_2$ (the block diagonal matrix M10), respectively. Note that the order of individuals in $\mathbf{V}_1$ and $\mathbf{V}_2$ must match, and should also match the order of individuals in the phenotype data frame.

5.2 Score tests

When performing score tests in GWAS, we need a fitted GLMM under the null hypothesis $H_0 : \beta = 0$ and a genotype file. We can construct score tests using theta, residuals (a vector of length n, the sample size), P (an $n \times n$ projection matrix) returned from the function **glmmkin**. Note that score tests require only vector/matrix multiplications and are much faster than Wald tests, which require fitting a new GLMM for each SNP. Score tests give the direction of effects but not effect size estimates. However, we can simply add score statistics and their variances from different studies to perform a meta-analysis. Here we provide a simple example of score tests using the plain text genotype file geno.txt:

```
> glmm.score.text(model0$res/model0$theta[1], model0$P, infile = "geno.txt",
+         outfile = "glmm.score.text.testoutfile.txt", infile.nrow.skip = 5,
+         infile.ncol.skip = 3, infile.ncol.print = 1:3, infile.header.print =
+         c("SNP", "Allele1", "Allele2"))
[1] "Computational time: 0.01 seconds"
[1] 0.01
```

The first argument in **glmm.score.text** is the residual vector (SCALED by the dispersion parameter estimate), and the second argument is the projection matrix P, both of which are returned from the null GLMM. The argument "infile" is the name (and path if not in the current working directory) of the plain text genotype file (or compressed files .gz and .bz2), and the argument "outfile" is the name of the output file. In this example genotype file, we have 5 comment lines to skip using "infile.nrow.skip". The first 3 columns contain information on SNP name and alleles, which we skip from the analysis using "infile.ncol.skip" but subsequently keep in the output file using "infile.ncol.print" to select the 1st, 2nd and 3rd columns. Corresponding column names in the output file can be assigned using "infile.header.print".

Alternatively, if your genotype information is saved in a PLINK binary file "geno.bed", you can use **glmm.score.bed** to perform score tests:

```
> glmm.score.bed(model0$res/model0$theta[1], model0$P, plinkfiles = "geno",
+         outfile = "glmm.score.bed.testoutfile.txt")
[1] "Computational time: 0.01 seconds"
[1] 0.01
```

Here "plinkfiles" is the prefix (and path if not in the current working directory) of the PLINK files (.bed, .bim and .fam). SNP information in the .bim file (in our example, "geno.bim") is carried over to the output file.

Both **glmm.score.text** and **glmm.score.bed** return the actual computation time in seconds from their function calls.

5.3 Wald tests

When performing Wald tests for candidate SNPs to get effect size estimates, we need the phenotype (and covariates) data frame, the matrices modeling the covariance structure of the random effects, and the genotype file. To perform Wald tests, we do not need fitting the null GLMM required in score tests using **glmmkin**. In the example below, we perform Wald tests for 4 candidate SNPs of interest and get their effect estimates:

```
> snps <- c("SNP10", "SNP25", "SNP1", "SNP0")
> glmm.wald.text(fixed = disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"), infile = "geno.txt", snps,
+         infile.nrow.skip = 5, infile.ncol.skip = 3, infile.ncol.print = 1:3,
+         infile.header.print = c("SNP", "Allele1", "Allele2"))
    SNP Allele1 Allele2   N         AF        BETA        SE      PVAL
1 SNP10       A       G 400 0.23250000 -0.13976670 0.1740090 0.4218502
2 SNP25       A       C 400 0.17500000 -0.02920747 0.1934447 0.8799866
3  SNP1       A       T 393 0.02544529  0.45660682 0.4909945 0.3523901
4  SNP0    <NA>    <NA>  NA         NA          NA        NA        NA
  converged
1      TRUE
2      TRUE
3      TRUE
4        NA
```

The syntax is a hybrid of **glmmkin** and **glmm.score.text**. Note that the argument "fixed" is a formula including the covariates but NOT the test SNPs. The argument "snps" is a character vector of the names of the test SNPs. The function **glmm.wald.text** returns a data frame with first columns copied from the genotype file using "infile.ncol.print" and names specified using "infile.header.print", followed by the sample size N, the allele frequency (AF) of the effect allele (Allele2 recommended, but you can also define Allele1 as the effect allele in your coded genotype file), effect size estimate BETA of the effect allele, standard error SE, Wald test P value PVAL, and an indicator for whether the GLMM is converged. Note that in the example above, SNP0 is not actually included in the genotype file, so all results are missing.

Alternatively, if your genotype information is saved in a PLINK binary file "geno.bed", you can use **glmm.wald.bed** to perform Wald tests:

```
> snps <- c("SNP10", "SNP25", "SNP1", "SNP0")
> glmm.wald.bed(fixed = disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"), plinkfiles = "geno", snps)
  CHR   SNP   cM  POS   A1   A2   N        AF        BETA        SE      PVAL
1   1 SNP10    0   10    G    A 400 0.7675000  0.13976670 0.1740090 0.4218502
2   1 SNP25    0   25    C    A 400 0.8250000  0.02920747 0.1934447 0.8799866
```

```
3    1  SNP1    0    1    T    A 393 0.9745547 -0.45660682 0.4909945 0.3523901
4 <NA>  SNP0 <NA> <NA> <NA> <NA>  NA         NA          NA        NA        NA
  converged
1      TRUE
2      TRUE
3      TRUE
4        NA
```

It returns a data frame with first 6 columns copied from the .bim file (in our example, "geno.bim"), followed by the sample size N, the allele frequency (AF) of A2 allele (the effect allele, note that A1 allele in .bim is coded 0 and A2 allele is coded 1), effect size estimate BETA of A2 allele, standard error SE, Wald test P value PVAL, and an indicator for whether the GLMM is converged.

6 Output

The score test functions **glmm.score.text** and **glmm.score.bed** generate a tab-delimited plain text output file. Here are the header and the first 5 rows of the example output file "glmm.score.text.testoutfile.txt" from **glmm.score.text**:

```
SNP     Allele1 Allele2 N       AF       SCORE   VAR     PVAL
SNP1    A       T       393     0.0254453       1.98499 4.55635 0.352407
SNP2    A       C       400     0.5      3.51032 46.3328 0.606059
SNP3    A       C       400     0.2075   -0.533396       30.6023 0.923186
SNP4    A       G       400     0.29875 -3.11494         40.5127 0.624566
SNP5    A       G       400     0.59375 -4.00133         42.2757 0.538289
...
```

The first columns are copied from the genotype file using "infile.ncol.print" with names specified using "infile.header.print". Results are included in 5 columns: the sample size N, the allele frequency (AF) of the effect allele (Allele2 recommended, but it is the user's choice: you can also define Allele1 as the effect allele in your coded genotype file), the score statistic SCORE of the effect allele, the variance of the score VAR, and score test P value PVAL.

Here are the header and the first 5 rows of the example output file "glmm.score.bed.testoutfile.txt" from **glmm.score.bed**:

```
CHR     SNP     cM      POS     A1      A2      N       AF
1       SNP1    0       1       T       A       393     0.974555
1       SNP2    0       2       A       C       400     0.5
1       SNP3    0       3       C       A       400     0.7925
1       SNP4    0       4       G       A       400     0.70125
1       SNP5    0       5       A       G       400     0.59375
...
SCORE       VAR     PVAL
-1.98499    4.55635 0.352407
3.51032     46.3328 0.606059
0.533396    30.6023 0.923186
3.11494     40.5127 0.624566
```

```
-4.00133    42.2757 0.538289
...
```

The first 6 columns are copied from the .bim file (in our example, "geno.bim"): the chromosome CHR, SNP name, genetic location cM, physical position POS, and alleles A1 and A2. Results are included in 5 columns: the sample size N, the allele frequency (AF) of A2 allele, the score statistic SCORE of A2 allele, the variance of the score VAR, and score test P value PVAL.

7 Advanced options

7.1 Alternative model fitting algorithms

By default we use the Average Information REML algorithm[4,5] to fit the GLMM in **glmmkin**, which is computationally efficient and recommended in most cases. However, there are also alternative model fitting algorithms:

```
method = "REML", method.optim = "Brent"
```

It maximizes the restricted likelihood using the derivative-free Brent method,[6] but only works when there is one matrix for the covariance structure of the random effects.

```
method = "ML", method.optim = "Brent"
```

It maximizes the likelihood using the Brent method.

```
method = "REML", method.optim = "Nelder-Mead"
```

It maximizes the restricted likelihood using the Nelder-Mead method,[7] however it is usually very slow in large samples.

```
method = "ML", method.optim = "Nelder-Mead"
```

It maximizes the likelihood using the Nelder-Mead method.

Note that the default algorithm is

```
method = "REML", method.optim = "AI"
```

A maximum likelihood version of Average Information algorithm is not available in **glmmkin**.

7.2 Changing model fitting parameters

By default we set the maximum number of iteration to 500 and tolerance to declare convergence to 1e-5:

```
maxiter = 500, tol = 1e-5
```

These parameters can be changed. When using the Brent method for maximizing the likelihood (or restricted likelihood), we specify the search range of the ratio of the variance component parameter τ_k over the dispersion parameter ϕ to be between 1e-5 and 1e5, and we divide the search region evenly into 10 regions on the log scale:

```
taumin = 1e-5, taumax = 1e5, tauregion = 10
```

These parameters can also be changed, but they are only effective when using the Brent method.

7.3 Missing genotypes

It is recommended to perform genotype quality control prior to analysis to impute missing genotypes or filter out SNPs with high missing rates. However, *GMMAT* does allow missing genotypes, and imputes to the mean value by default. Alternatively, missing genotypes can be omitted from the analysis using

```
missing.method = "omit"
```

If using a plain text (or compressed .gz and .bz2) genotype file, missing genotypes should be coded as "NA". If you have missing genotypes coded in a different way, you can specify this in the argument "infile.na".

7.4 Reordered genotypes

The genotype file (either a plain text file or a PLINK binary file) can include more individuals than in the phenotype and covariates data frame or the matrices, and they can be in different orders. *GMMAT* handles this issue using an argument "select" in both score test and Wald test functions. For example, if the order of individuals in your genotype file is A, B, C, D, but you only have 3 individuals in the phenotype and covariates data frame and the matrices (with order C, A, B), then you can specify

```
select = c(2, 3, 1, 0)
```

to reflect the order of individuals. Note that since individual D is not included in the phenotype and covariates data frame or the matrices, its order is assigned to 0. The length of the vector must match the number of individuals in your genotype file. The order of individuals in the phenotype and covariates data frame and the covariance structure matrices must be the same.

7.5 Other options

By default, genotypes are centered to the mean before the analysis. You can turn this feature off by specifying

```
center = FALSE
```

in both score test and Wald test functions to use raw genotypes.

If your genotype file is a plain text (or a compressed .gz and .bz2 file), and you want to read in fewer lines than all lines included in the file, you can use the "infile.nrow" argument to specify how many lines (including lines to be skipped using "infile.nrow.skip") you want to read in. By default the delimiter is assumed to be a tab, but you can change it using the "infile.sep" argument.

In the score test, by default 100 SNPs are tested in a batch. You can change it using the "nperbatch" argument, but the computational time can increase substantially if it is either too small or too large, depending on the performance of your system.

If you perform Wald tests and use a plain text (or a compressed .gz and .bz2) file, and your SNPs are not in your first column, you can change "snp.col" in **glmm.wald.text** to indicate which column is your SNP name.

8 Version

8.1 Version 0.6 (October 12, 2015)

Initial public release of *GMMAT*.

9 Contact

Please refer to the R help document of *GMMAT* for specific questions about each function. For comments, suggestions, bug reports and questions, please contact Han Chen (hanchen@hsph.harvard.edu). For bug reports, please include an example to reproduce the problem without having to access your confidential data.

10 Acknowledgments

We thank Dr. Chaolong Wang and Dr. Brian Cade for comments and suggestions on *GMMAT* and the user manual.

References

[1] Breslow, N. E. and Clayton, D. G. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* **88,** 9–25 (1993).

[2] Chen H., Wang C., Conomos M. P., Stilp A. M., Li Z., Sofer T., Szpiro A. A., Chen W., Brehm J. M., Celedón J. C., Redline S. S., Papanicolaou G. J., Thornton T. A., Laurie C. C., Rice K. and Lin X. Control for Population Structure and Relatedness for Binary Traits in Genetic Association Studies Using Logistic Mixed Models. Submitted.

[3] Zhou, X. and Stephens, M. Genome-wide efficient mixed-model analysis for association studies. *Nature Genetics* **44,** 821–824 (2012).

[4] Gilmour, A. R., Thompson, R. and Cullis, B. R. Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics* **51,** 1440–1450 (1995).

[5] Yang, J., Lee, S. H., Goddard, M. E. and Visscher, P. M. GCTA: a tool for genome-wide complex trait analysis. *The American Journal of Human Genetics* **88,** 76–82 (2011).

[6] Brent, R. P. Chapter 4: An Algorithm with Guaranteed Convergence for Finding a Zero of a Function, Algorithms for Minimization without Derivatives, Englewood Cliffs, NJ: Prentice-Hall, ISBN 0-13-022335-2 (1973).

[7] Nelder, J. A. and Mead, R. A simplex algorithm for function minimization. *Computer Journal* **7,** 308–313 (1965).